Learning analytics in computer programming courses

Edna Chaparro[1], Felipe Restrepo-Calle[1] and Jhon Jairo Ramírez-Echeverry[1]

[1]*Universidad Nacional de Colombia, Bogotá, Colombia*

Abstract

In recent years, learning analytics has emerged as one of the most important fields on the future of education. In the context of programming courses, the applications of learning analytics show high efficacy in giving directions for interventions, which help to promote better learning methods. However, few investigations have considered complex datasets where there are heterogeneous student's groups, letting out differential factors that can influence in the learning process. The main objective of this work is determining the relations between the measurements and metrics of the learning process with the academic performance of the computer programming students of the National University of Colombia. We apply a quantitative non-experimental methodological design, using as source of information the records of 2 years of student's interactions with an educational platform of automatic grading and feedback use in the course. In total 38 variables are considered in this work, that include the number of submissions, the results of each submission, software metrics, and the use rates of the tools available in the platform. The results show that the number of submissions, three types of results/verdicts, two verdict's rates, and one software metric have a positive correlation with the academic performance. Moreover, the runtime error rate, and the use of a good practices verification tool (i.e., Linter) have a negative correlation with the final performance of the students.

Keywords

Learning Analytics, Computer Programming, Quantitative data, Correlational analysis

1. Introduction

The last decade has seen an increase in the use of technology in educational environments with tools such as computers, electronic boards, virtual environments and learning management systems. Consequently, the data collected during the learning process have grown exponentially, along with their potential to generate knowledge about possible factors for academic success [1, 2]. In this sense, the information generated can be used to guide teachers, institutions, and students in making decisions related to learning, teaching, and educational administration [3, 4]. Data analysis in the educational area is known as learning analytics and is considered the future of education, especially in the context of higher education [3]. Learning analytics is based on the principles of traditional educational research, but it leverages innovations such as the collection of new forms of digital data and uses advanced computational analysis techniques

LALA'21: IV Latin American Conference on Learning Analytics, October 19–21, 2021, Arequipa, Perú

edchaparroa@unal.edu.co (E. Chaparro); ferestrepoca@unal.edu.co (F. Restrepo-Calle); jjramireze@unal.edu.co (J. J. Ramírez-Echeverry)
0000-0002-6147-6606 (E. Chaparro); 0000-0003-4226-1324 (F. Restrepo-Calle); 0000-0002-6499-1785 (J. J. Ramírez-Echeverry)

from data science and artificial intelligence [5, 6]. In the context of computer programming courses, learning analytics has been used to achieve different objectives. Several efforts seek to build models to automatically detect students at risk of failing a course [7, 8]. Others propose to track course progress through visualizations of statistics and metrics representing student behaviors [9], and to generate personalized feedback to students in programming courses [10].

However, Ferguson [11] has identified that research in learning analytics lacks methods that use a wide range of datasets. This coincides with one of the challenges presented by Schmitz et al. [12], who find a lack of educational investigations that considers diversity of learners. This challenge is due to the fact that advances in education facilitates the inclusion of relatively new or different groups of students than the traditional ones [12]. Thus, in this work we propose to answer the following research question: What are the relationships between data generated from students' interactions with an educational platform and their academic performance in a computer programming course when considering different groups of students in the dataset? Thus, the objective of this work is to determine the existing relationships between measurements and metrics from the learning processes of computer programming students at the National University of Colombia and their academic performance. The methodology used is descriptive, correlational, quantitative and non-experimental.

2. Related work

One of the areas where learning analytics has gained importance is computer science [7]. In the context of computer programming courses, one of the most frequent objectives is the prediction of student behavior. For example, Azcona et al. [7] propose a model to automatically detect students at risk of failing a Python computer programming course and based on the predictions, provide personalized feedback. Other investigations seek to generate mechanisms for effective course monitoring with visualizations of statistics and metrics that represent the students' behavior. For example, Shen et al. [9], propose a heat map that visualizes the intensity of student access to educational resources and activities in an introductory Python MOOC. Additionally, through social network analysis, examines similarities and differences in access patterns. Another example is the set of visualizations proposed by Leony et al. [13] in a C programming course, aiming to inform teachers about students' emotions based on the interaction of students with an educational technology tool. Finally, other researches aim to generate personalized feedback to students. For example, Lu et al. [10] apply learning analytics to identify students who need immediate intervention in a Python MOOC, and based on this information, teachers build adaptive learning guides that are applied in an experimental group and compared with a control group.

3. Methodology

The learning analytics' methodology of this work has a quantitative approach with a descriptive and correlational scope. The type of research design is non-experimental, since the data to be used have been collected without modifying the variables of the context. The methodology consists of 3 sequential phases: 1) data preparation, 2) data transformation, and 3) data analysis.

3.1. Phase 1: Data preparation

Data collection. First, the location and format of the available data is identified. Then, the data of the students' learning process is gathered through a data management and analysis tool.

Data consolidation. A consolidation process is necessary, since the raw data is often disaggregated. Subsequently, the most appropriate data structures are identified for the organization and manipulation of the consolidated data.

Data cleaning. The variables present in the consolidated dataset which gives useful information about the learning process are identified. Also, the variables that are not related to the objective of the study or those that have data quality problems are discarded.

3.2. Phase 2: Data transformation

Identification of measurements. Firstly, a literature review of measurements and metrics used in educational research in computer programming courses should be performed. Subsequently, identify which of the measurements found in the literature are present in the dataset and which can be used to build suitable metrics.

Metrics design. From the results found on metrics in the literature, the equations needed for the estimation of metrics are established.

Calculation of metrics. The equations proposed in the metrics design stage are applied in the data management tool and the values obtained are stored for later analysis.

3.3. Phase 3: Data Analysis

Exploratory data analysis. Based on the previous measurements and metrics, an exploratory univariate analysis is performed to identify behaviors and trends. For this, descriptive statistics (arithmetic mean, dispersion measures, skewness, etc.) and visualizations (box plots, histograms, etc.) are used.

Data analysis and modeling techniques. The relationships between metrics and measurements are identified through correlations or regression analysis. Additionally, supervised or unsupervised machine learning techniques can be applied if the objective of the work is to obtain classifications, regressions or groupings of data.

Discussion of results. The results obtained are compared with similar findings of other studies. Then, the behaviors found are interpreted and the implications in the context of the research are specified.

Conclusions. The summary of the findings is presented along with its limitations and the possible threats to validity. Finally, future work is proposed based on the results of the study.

4. Results

Results obtained by applying the proposed methodology to the 2-year (between 2019 and 2020) history of the Computer Programming course at the National University of Colombia are presented below.

4.1. Phase 1: Data Preparation

Data collection. Data were collected from the interaction of students with the educational platform UNCode used for the automatic evaluation of programming exercises of class activities [14]. UNCode allows students to submit multiple program attempts (source code or Jupyter notebooks) built to solve programming tasks. For each solution attempt, the platform stores the program file, date and time of the submission. In addition, it offers automatic feedback through verdicts related to errors in the syntax, semantics and efficiency of the program. Also, it offers a numerical grading, depending on the test cases the designed program solved. Within UNCode there is a set of learners' support tools, such as: syntax highlighting, code auto-completion, linter (suggestions of good programming practices), visualization of code execution, custom tests and reports of grades [15]. The raw data was stored in a MongoDB database and it was exported to CVS (Comma Separated Values) files. The programming language Python was used, with its specialized libraries for data analysis and visualization (*pandas*, *matplotlib*, *seaborn*).

Data consolidation. The exported files in CSV format for each course were stored in a shared folder in Google Drive. The data was organized in separate folders per course where there are files with the list of student usernames, the records of the submissions made per activity, and the final grades. In addition, each course contains a folder with the files submitted, organized by student and by activity.

Data cleaning. The stored data includes all the courses of the university that use UNCode; for this reason, it contains data from courses other than computer programming. Therefore, a screening process is made by selecting only the programming courses (22 groups). Then, a filtering is performed within each course for activities and students with insufficient number of submissions. The post-filtering dataset contains 1352 activities and 735 students.

4.2. Phase 2: Data Transformation

Identification of measurements. Table 1 specifies the 15 measurements that are considered of interest in the research, which are classified into four categories: 1) data related to the attempts of solution of programming assignments made by each student, 2) the verdicts obtained in each solution attempt, 3) the use of the platform tools, and 4) numerical grading. Additionally, the source code files sent as solutions to the programming tasks are used to obtain characteristics of the students' programs (*software metrics*).

Metrics design. Based on the identified measures, 24 metrics were designed, divided into three categories (verdicts, tool usage and software metrics). See Table 2. First, the verdicts' rates

Table 1
Measurements considered in the dataset

Category	Measurements	Description
Submissions	*Total_Submissions*	Number of attempts submitted per student.
	Duration_of_Submission	Average time spent by students between submission attempts.
Verdicts	*Accepted*	Number of solutions with correct answers.
	Wrong_Answer	Number of solutions with incorrect answers.
	Compilation_Error	Number of submitted attempts that fail to compile.
	Runtime_Error	Number of attempts that succeed in compiling but fail during execution.
	Time_Limit_Exceeded	Number of attempts that take too long to execute.
	Memory_Limit_Exceeded	Number of attempts that exceed the memory available for execution.
	Output_Limit_Exceeded	Number of attempts that exceed the expected program output size.
Tool usage	*Python_Tutor*	Number of logged accesses to the Python tutor tool that allows visualization step-by-step execution of a program.
	Custom_Input	Number of registered accesses to the Custom input tool where students perform custom tests on their programs.
	Linter	Number of registered accesses to the Linter tool, which highlights syntax and style problems in the source code.
	User_Statistics	Number of registered accesses to the interactive dashboard to report on students' individual statistics.
	Multiple_Languages_Code	Number of accesses to the Multiple Languages tool that allows submission in different programming languages.
Academic performance	*uncode_grade*	Weighted average of grades of the activities performed by students in UNCode.

obtained for each type of verdicts in Table 1 are defined. These rates are the percentage of one type of verdict with respect to the total number of verdicts obtained by the student. The second category is the usage rates of each tool in Table 1. The usage rates are defined as the percentage of the number of accesses to a single tool in relation to the total number of accesses to all available tools (these two categories of rates are defined with the objective of quantifying the verdicts and tools of greater or lesser use by students).The third category are software metrics, which represent specific characteristics of the programs built by the students (see Table 3).

Calculation of metrics. For each student the 7 verdict rates and the 5 tool usage rates are calculated by applying the equations in Table 2. Then, the software metrics are calculated using

Table 2
Metrics based on the verdict measures obtained and tool usage

Category	Metric	Equation
Verdicts	*Success_rate*	$\frac{Accepted}{\sum_i Veredicts_i} \cdot 100$
	Error_rate_Wrong_Answer	$\frac{Wrong\_Answer}{\sum_i Veredicts_i} \cdot 100$
	Error_rate_Compilation_Error	$\frac{Compilation\_Error}{\sum_i Veredicts_i} \cdot 100$
	Error_rate_Runtime_Error	$\frac{Runtime\_Error}{\sum_i Veredicts_i} \cdot 100$
	Error_rate_Time_Limit_Exceeded	$\frac{Time\_Limit\_Exceeded}{\sum_i Veredicts_i} \cdot 100$
	Error_rate_Memory_Limit_Exceeded	$\frac{Memory\_Limit\_Exceeded}{\sum_i Veredicts_i} \cdot 100$
	Error_rate_Output_Limit_Exceeded	$\frac{Output\_Limit\_Exceeded}{\sum_i Veredicts_i} \cdot 100$
Tool usage	*Python_Tutor_rate*	$\frac{Python\_Tutor}{\sum_i Tools_i} \cdot 100$
	Custom_input_rate	$\frac{Custom\_input}{\sum_i Tools_i} \cdot 100$
	Linter_rate	$\frac{Linter}{\sum_i Tools_i} \cdot 100$
	User_Statistics_rate	$\frac{User\_Statistics}{\sum_i Tools_i} \cdot 100$
	Multiple_Languages_Codes_rate	$\frac{Multiple\_Languages\_Code}{\sum_i Tools_i} \cdot 100$

Table 3
Software metrics from source code files submitted by students.

Category	Metric	Description/Equation
Software metrics	Lines of code (*NLOC*)	Number of lines of source code excluding comments.
	Tokens_count	Num. reserved words of the programming language used in program.
	Ciclomatic complexity (*G*)	Number of decision blocks contained in the code plus one.
	Program vocabulary (*n*)[a,b]	$n = n_1 + n_2$
	Program length (*N*)[c,d]	$N = N_1 + N_2$
	Calculated program length (L)	$L = n_1 \cdot log_2(n_1) + n_2 \cdot log_2(n_2)$
	Volume (*V*)	$V = N \cdot log_2(n)$
	Difficulty (*D*)	$D = \frac{n_1}{2} \cdot \frac{N_2}{n_2}$
	Effort (*E*)	$E = D \cdot V$
	Time required to program (*T*)	$T = E/18$
	Number of delivered bugs: (*B*)	$B = V/3000$
	Maintainability Index (*MI*)[e,f,g]	Measure of how easy to support and change the source code is (0-100).

[a] n_1 : The number of distinct operators.
[b] n_2 : The number of distinct operands.
[c] N_1 : The total number of operators.
[d] N_2 : The total number of operands.
[e] $MI = max[0.1 \cdot \sqrt{171 - 5.2 \cdot Ln(V) - 0.23 \cdot G - 16.2 \cdot Ln(SLOC) + 50 \cdot sin(2.4 \cdot C)}/171]$
[f] $SLOC$: The number of source lines of code.
[g] C : The percent of comment lines converted to radians.

the Python libraries *lizard*[1] and *radon*[2].

4.3. Phase 3: Data analysis

Exploratory data analysis. Figure 1 summarizes the most important results of the univariate analysis. In the case of total submissions made each student (top-left in Figure 1) on average makes ($\overline{x}$) 176.6 attempts with a standard deviation (s) of 120.8. Regarding the average time between submissions (top-right in Figure 1) the $\overline{x}$ is 7.1 h with a s of 19.8 h. On the other hand, the plot of verdicts (bottom-left in Figure 1) evidences that the verdicts with the highest rates are *Error_rate_Wrong_Answer* and *Success_rate* with 48.9% and 31.7%, respectively. Regarding the tool usage rates, the total rates graph (bottom-right in Figure 1) shows that the most used tool by students is *Custom_input_rate* being 65.0% of the total accesses. This is followed by *Python_Tutor_rate* and *Multiple_Languages_Codes_rate* with percentages of 17.7% and 12.0%, respectively. Table 4 summarizes the statistical values describing the software metrics calculated based on the programs built by the students. Besides that, Figure 2 shows the distribution of the (*uncode_grades*) in each group[3]. The total $\overline{x}$ is 4.1 with a s of 0.9.

Data analysis and modeling techniques. To identify which measurements or metrics are significantly related to the students' academic performance, a correlation analysis is performed. Considering that all variables are continuous, Pearson's correlation coefficient is used. Figure 2

[1]It is used to quantify the *NLOC* and the *Tokens_count*.
[2]It is used to calculate the cyclomatic complexity (*G*), the maintainability index (*MI*) and the Halstead metrics (Vocabulary, Length, Calculated length, Volume, Difficulty, Effort, Time to program, and Number of delivered bugs.)
[3]The range of the values are between 0.0 and 5.0 with a minimum passing grade was 3.0

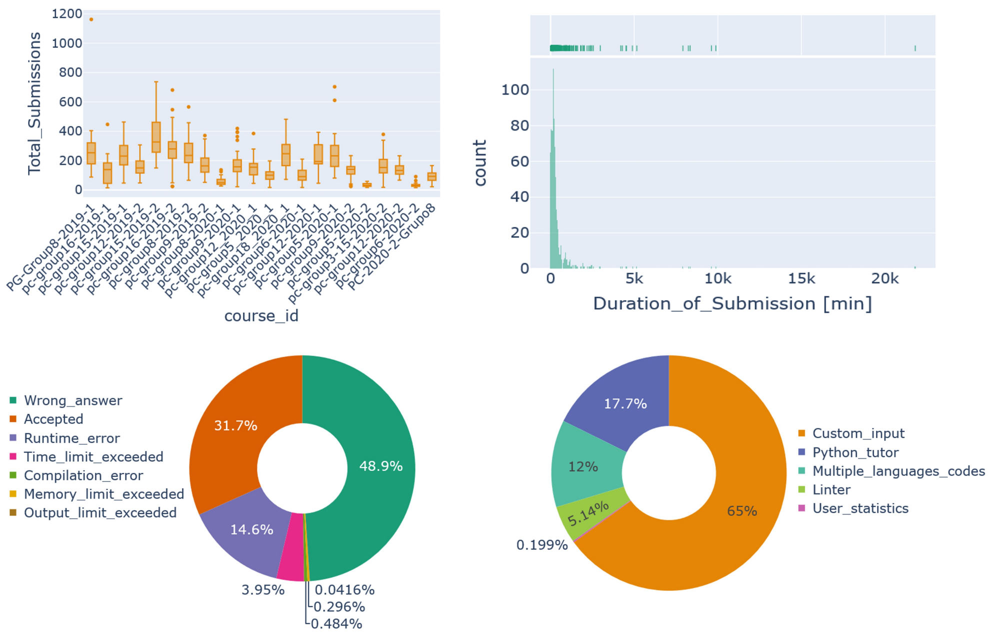

Figure 1: Results of the calculated measurements and metrics. a) Top-Left: Distribution of *Total_Submissions* per student in each course. b) Top-Right: Histogram of average time between attempts (*Duration_of_Submission*) for each student. c) Bottom-Left: Total verdict rates obtained in all courses. d) Bottom-Right: Total rates of tool usage across all courses.

Table 4
Descriptive statistics of software metrics.

Metric	Arithmetic mean	Standard deviation	Minimum	Maximum
Code lines (*NLOC*)	20.7	9.4	2.0	92.1
Tokens_count	150.2	66.1	25.0	565.4
Cyclomatic complexity (*G*)	7.8	4.4	1.0	42.2
Program vocabulary (*n*)	20.8	10.0	3.0	82.0
Program length (*N*)	42.0	25.7	3.0	225.2
Calculated program length (L)	88.3	66.3	2.0	534.7
Volume (*V*)	214.6	117.5	4.8	1740.8
Difficulty (*D*)	3.7	1.3	0.5	12.6
Effort (*E*)	1290.0	1603.8	2.4	20181.8
Time required to program (*T*)	71.7	89.1	0.1	1121.2
Number of delivered bugs (*B*)	0.1	0.1	0.002	0.6
Maintainability Index (*MI*)	60.9	6.4	40.1	86.1

illustrates 21 measurements and metrics that have a statistically significant correlation (*p-valor* ≤ 0.05) with the academic performance.

The variable with the highest positive correlation is *Accepted* with coefficient of 0.43. This value is followed by *Success_rate* and *Total_Submissions* with coefficients of 0.25 and 0.24, respectively. These are expected results since a student with a high number of correct answers, success rate and attempts made is a student who is successful in solving the assignments. Subsequently, the variables with correlations less than 0.20 and greater than 0.15 correspond to *Wrong_Answer*, *Time_Limit_Exceeded*, *Tokens_count*, *Custom_Input_rate*, and *NLOC*. The num-

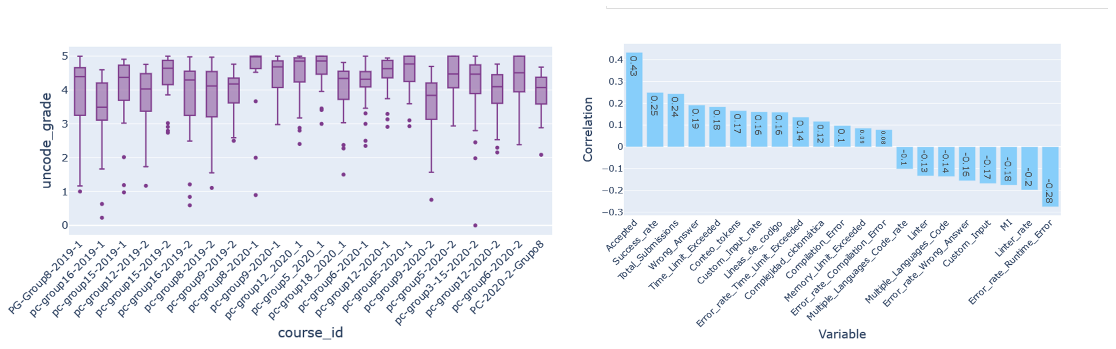

Figure 2: a) Left: Distribution of academic performance by group. b) Right: Variables with significant correlations with the academic performance.

ber of tokens, lines of code and Custom Input usage rate are metrics that were expected to be positively correlated, since as the student designs high content programs and is able to perform custom testing of the built code, his/her academic performance could be better. However, the number of incorrect answers and time limit exceeded errors were not expected to be in this group, because these verdicts indicate errors in the proposed solution.

In contrast, the variable with the highest negative correlation is *Error_rate_Runtime_Error* with -0.28. This metric was expected to be negatively correlated, given that a high runtime error rate represents that a large number of the non-executed attempts. The variables with correlations between -0.20 and -0.15 correspond to *Linter_rate*, *MI*, *Custom_Input* and *Error_rate_Wrong_Answer*. In these results, it is rare that the rate of use of Linter and the accesses to Custom Input have negative correlations, since these tools are designed to facilitated students to build their solutions. The maintainability index and incorrect answer rate were expected to be negatively correlated, since a program with a high maintainability index indicates a low level of programming skills and a high incorrect answer rate means that most of the attempts made were unsuccessful.

5. Discussion

Regarding the research question posed: what are the existing relationships between the data generated from students' interactions with an educational platform and their academic performance in a computer programming course when considering different groups of students?, some of the findings found in the results are discussed below. The total number of submissions made per student has a positive correlation with the final grade in the course. This may indicate that students with high performance use the platform as a source of feedback to improve the solutions constructed, making multiple attempts. This result agrees with the findings of Zacharis et al. [16] who found a weak correlation (from 0.20 to 0.39) between number of activities delivered and the final course grade. The positive correlations of the number of correct answers and the success rate indicate that the students have possibly acquired sufficient knowledge to successfully solve the course activities and this is reflected in their performances.

In contrast, the positive correlation of the number of incorrect answers, the number of memory limit errors exceeded, along with the number of verdicts and rates of time limit exceeded and compilation error may indicate that these verdicts provide sufficient feedback to guide the student to solve the possible errors present in the constructed solution. However, the rate of incorrect answer and execution error has negative correlations. In the case of high incorrect answers rate it may indicate a lack of understanding of the objective of the activity; with respect to the execution error rate, this may indicate the students' difficulty to obtain effective feedback for his learning process.

6. Conclusions and future work

This work applies a non-experimental quantitative methodological design of learning analytics, consisting of three main phases: data preparation, transformation and analysis, applied to 2 years of historical data of a computer programming course. This allowed finding relationships between measurements and metrics of the learning process and the final students' performance of a computer programming course. The results show that the number of submissions, five verdicts(correct answer, incorrect answer, time limit exceeded, memory limit exceeded and compilation error), three verdict rates (correct answer, time limit exceeded and compilation error), the rate of use of Custom Input, and four software metrics (number of tokens, number of lines of code, cyclomatic complexity and difficulty) have a significant positive correlation with the academic performance of the students. This indicates that students with high performance may be able to use the error verdicts obtained as a source of formative feedback and that the construction of programs of high content, length and complexity is related to better academic performance. The variables with negative correlations are two verdict rates (incorrect answer and execution error), three tools' accesses (linter, multiple languages code and custom input), two tools' usage rates (linter and multiple languages code) and the maintainability index. These findings may indicate the need for reinforcement in some aspects of the course, such as, clarity in the objective of the activities, the applicability of the feedback offered by some tools (i.e., linter that focuses on the verification of good programming practices) and promote the learning of generalizable program construction skills.

The main limitation of this study is related to the quantitative approach, since the results are limited to evidencing relationships between measurements and metrics, but it is not possible to identify the causes of the found behaviors. As future work, the constructed dataset can be used to identify which metrics are most relevant and to track students and build a predictive model of final academic performance. Qualitative data will also be considered within the analysis in future works.

References

[1] G. Siemens, Learning analytics: The emergence of a discipline, American Behavioral Scientist 57 (2013) 1380–1400. doi:10.1177/0002764213498851.

[2] R. S. Baker, P. S. Inventado, Educational Data Mining and Learning Analytics, Springer New York, New York, NY, 2014, pp. 61–75. doi:10.1007/978-1-4614-3305-7_4.

[3] P. Long, G. Siemens, Penetrating the fog: Analytics in learning and education, EDUCAUSE Review 46 (2011) 31–40. URL: https://er.educause.edu/articles/2011/9/penetrating-the-fog-analytics-in-learning-and-education.

[4] V. S. Kumar, Kinshuk, T. S. Somasundaram, D. Boulanger, J. Seanosky, M. F. Vilela, Big Data Learning Analytics: A New Perpsective, Springer Berlin, 2015, pp. 139–158. doi:10.1007/978-3-662-44659-1_8.

[5] S. Society for Learning Analytics Research, What is learning analytics?, 2021. URL: https://www.solaresearch.org/about/what-is-learning-analytic.

[6] M. D. Pistilli, J. E. Willis, J. P. Campbell, Analytics Through an Institutional Lens: Definition, Theory, Design, and Impact, Springer New York, 2014, pp. 79–102. doi:10.1007/978-1-4614-3305-7_5.

[7] D. Azcona, I.-H. Hsiao, A. F. Smeaton, Detecting students-at-risk in computer programming classes with learning analytics from students' digital footprints, User Modeling and User-Adapted Interaction 29 (2019) 759–788. doi:10.1007/s11257-019-09234-7.

[8] J. Lagus, K. Longi, A. Klami, A. Hellas, Transfer-learning methods in programming course outcome prediction, ACM Trans. Comput. Educ. 18 (2018). doi:10.1145/3152714.

[9] H. Shen, et al, Understanding learner behavior through learning design informed learning analytics, in: Proc Conf Learning @ Scale, ACM, NY, USA, 2020, p. 135–145. doi:10.1145/3386527.3405919.

[10] O. H. T. Lu, J. C. H. Huang, A. Y. Q. Huang, S. J. H. Yang, Applying learning analytics for improving students engagement and learning outcomes in an moocs enabled collaborative programming course, Interactive Learning Environments 25 (2017) 220–234. doi:10.1080/10494820.2016.1278391.

[11] R. Ferguson, Learning analytics: drivers, developments and challenges, International Journal of Technology Enhanced Learning 4 (2012) 304–317. doi:10.1504/IJTEL.2012.051816.

[12] M. Schmitz, et al, Opportunities and challenges in using learning analytics in learning design, in: E. Lavoué, et al (Eds.), Data Driven Approaches in Digital Education, Springer International Publishing, 2017, pp. 209–223. doi:10.1007/978-3-319-66610-5_16.

[13] D. Leony, et al, Provision of awareness of learners emotions through visualizations in a computer interaction environment, Expert Systems with App 40 (2013) 5093–5100. doi:10.1016/j.eswa.2013.03.030.

[14] F. Restrepo-Calle, J. Ramírez-Echeverry, F. A. González, Using an interactive software tool for the formative and summative evaluation in a computer programming course: an experience report, Global Journal of Engineering Education 22 (2020) 174–185. URL: http://www.wiete.com.au/journals/GJEE/Publish/vol22no3/06-Echeverry-J.pdf.

[15] F. Restrepo-Calle, J. Ramírez-Echeverry, F. Gonzalez, UNCode: Interactive System for Learning and Automatic Evaluation of Computer Programming Skills, in: EDULEARN18, 10th Int Conf on Education and New Learning Technologies, IATED, 2018, pp. 6888–6898. URL: http://dx.doi.org/10.21125/edulearn.2018.1632. doi:10.21125/edulearn.2018.1632.

[16] N. Z. Zacharis, A multivariate approach to predicting student outcomes in web-enabled blended learning courses, The Internet and Higher Education 27 (2015) 44–53. doi:10.1016/j.iheduc.2015.05.002.